Event Based Sampling

Lab Exercises - Chapter 12

ACTIVITY 12-1: CONDUCTING A SYSTEM-WIDE ANALYSIS

In this activity you perform a system-wide analysis to see how well the programs are running on your machine. You can run this activity on Linux or Windows.

1. Start Amplifier XE GUI from the command line:

   amplxe-gui

2. Create a new Chapter 12 project by selecting File | New | Project. In the Project Properties dialog, select Profile System from the Target tab.

3. Carry out a Hotspot Analysis by selecting File | New | Analysis... | Algorithm Analysis | Lightweight Hotspots.

4. Stop the data collector after about ten seconds or so.

5. Explore the results and find the following:

   - The application with the largest CPI
   - Any fields highlighted in pink

ACTIVITY 12-2: CONDUCTING A LIGHTWEIGHT HOTSPOTS ANALYSIS

In this activity you look for the hotspots in the application memory.exe. You can run this activity on Linux or Windows.

1. Build the matrix.cpp application from Listing 12-3 (at the end of the chapter):

LINUX

```
g++ matrix.cpp -g -O2 -o matrix.exe
```

WINDOWS

```
cl matrix.cpp/Zi /O2 -o matrix.exe
```

2. In the Amplifier Project you already have open (from Activity 12-1), start a new Amplifier analysis and choose Lightweight Hotspots (File | New | Analysis... | Algorithm Analysis | Lightweight Hotspots).

3. In the Project Properties Target tab, select Launch Application.

4. Start the analysis by pressing the Start button.

5. When the analysis is complete, find the biggest hotspot in from the Bottom-up tab.

6. Double-click the hotspot and find the following in the source code:

   - The C line taking up the most time
   - The assembler instruction taking up the most time
   - The CPI rate of the bottleneck

7. Look in the Summary window and confirm that the hotspot you discovered is the same as the one mentioned on the summary page. (Figure 12-4 shows the summary button that you can use to display the summary page.)

ACTIVITY 12-3: CONDUCTING A GENERAL EXPLORATION ANALYSIS

In this activity you run a General Exploration analysis on matrix.cpp and discover the underlying hardware issue. You can run this activity on Linux or Windows.

1. If you haven't already done so, build the matrix.cpp application from Listing 12-3:

LINUX

```
g++ matrix.cpp -g -O2 -o matrix.exe
```

WINDOWS

```
cl matrix.cpp/Zi /O2 -o matrix.exe
```

2. Start new Amplifier analysis and choose General Exploration.

   File | New | Analysis...

3. In the Project Properties Target tab, select Launch Application and make sure the application to launch is matrix.exe.

4. In the list of prebuilt analysis types, chose < *CPU architecture* > | General Exploration, where *CPU architecture* will be one of the following:

   - Advanced Intel Core 2 Processor Family Analysis
   - Advanced Intel Microarchitecture Code Name Nehalem Analysis
   - Advanced Intel Microarchitecture Code Name Sandy Bridge Analysis
   - Advanced Intel Atom Processor Analysis

   Amplifier XE will only let you choose the analysis type that fits your CPU. If you choose an invalid option, the message This analysis type is only defined for processors based on... will be displayed, and the Start button will be disabled.

5. Start the analysis by selecting the Start button.

6. When the analysis is complete, look at the summary page. You should see some fields highlighted in pink.

7. Browse the Bottom-up view and find the biggest hotspot.

8. Navigate to the source code by double-clicking the hotspot and identify which source line is causing the problem.

ACTIVITY 12-4: OPTIMIZING THE APPLICATION

In this activity you fix the hardware issue identified in Activity 12-3. In the first two parts, optimization is achieved by modifying the code. In the last part, the optimizations are achieved automatically by using the Intel compiler. You can run this activity on Linux or Windows.

Implementing Loop Swapping

1. Copy the contents of Listing 12-3 into a file named swapped.cpp.

2. Swap the three nested loops so that the sequence is as follows:

```
// do the matrix calculation c = a * b
  for (i = 0; i < N; i++) {
    for (k=0; k<N; k++) {
      for (j=0; j<N; j++) {
        c[N*i+j] += a[N*i+k] * b[N*k+j];
      }
    }
  }
```

3. Build swapped.cpp:

LINUX

```
g++ swapped.cpp -g -O2 -o swapped.exe
```

WINDOWS

```
cl swapped.cpp /Zi /O2 -o swapped.exe
```

4. Make sure the Project Properties page points to swapped.exe.
5. Start a new Amplifier analysis and choose General Exploration (following the same instructions as Step 4 of Activity 12-3).
6. Start the analysis by pressing the Start button.
7. When the analysis is complete, look in the summary window. There should be fields highlighted in pink. (The results may vary, depending on what PC you are running on. You may even find that there are no highlighted fields in your results.)
8. Browse to the Bottom-up view and find the biggest hotspot.
9. Navigate to the source code by double-clicking the hotspot and confirm that the hardware issues from the summary page are associated with that line.

Using SSE Instructions to Speed Up the Code

10. Copy the contents of Listing 12-3 into a file named sse.cpp.
11. Replace the main function with the code in Listing 12-4.
12. Build sse.cpp:

LINUX

```
g++ sse.cpp -msse3 -g -O2 -o sse.exe
```

WINDOWS

```
cl sse.cpp /Zi /O2 -o sse.exe
```

13. Repeat steps 4 to 9, looking at sse.exe.

Using the Intel compiler

14. Rebuild the original matrix.cpp (which you created in step 1 of this activity) with the Intel compiler:

LINUX

```
icc matrix.cpp -g -O2 -o intel.exe
```

WINDOWS

```
icl matrix.cpp /Zi /O2 -o intel.exe
```

15. Repeat steps 4 to 9, looking at intel.exe.